Based on Spark SQL Design and Implementation of distributed full-text retrieval framework based on

Jiaxin Lan
Harbin Architecture University, Shijiazhuang, Hebei Province, 050000

Abstract: With the deepening of Information Technology, Big Data has created great value in various fields.. Storage and fast analysis of massive data has become a new challenge.. Traditional relational database due to Performance, Disadvantages of scalability and high price, Difficult to meet the storage and analysis needs of big data. Spark SQL is based on the Big Data Processing Framework, Data analysis tools, Currently supported TPC-DS Benchmark, Become an alternative solution to traditional data warehouse under the background of big data. Full-text retrieval as an effective way of text search, Can be used in conjunction with general query operations, Provide richer query and Analysis

Keywords: Operation; SparkSQL; Simple query operation; Full-text retrieval; traditional business migration;

1. Introduction

In a relational database, Full-text retrieval is an important index to measure the usability and functional completeness of Database. Full-text retrieval matches keyword and stored document data, Information Retrieval Technology for several documents with high correlation degree. In many relational databases, Such as MySQL, SQL Server, All have full-text retrieval capabilities.

However, SparkSQL as an alternative system to traditional data warehouse, Full-text retrieval is not supported SQL Statement and Its parallelization. Existing

To meet the requirements of traditional business migration and existing business for retrieval, This paper designs and realizes SparkSQL Distributed full-text retrieval framework. The main contributions of this paper are as follows:


2) A parallel method for full-text retrieval tasks is proposed, Including index build and query Parallelism.

3) Two retrieval optimization schemes are proposed. Two different schemes focused on performance optimization and storage optimization. Each scheme includes index storage and restore of original table data. Optimized scenario for storage, Proposed time complexity O (N) Connection Algorithm Between query results and original table data. Performance of frames using large data sets, Scalability is evaluated, Compared with the traditional relational database. Experiments show that, Compared to traditional relational database, Under two retrieval Optimization Strategies, Build time of the Framework Index, Query time is the same as the traditional database 0.5%/1%, 10%/0.6%, Index storage is reduced 55.0%.
This article describes the Big Data Query Language and framework, SQL Execution Plan build and optimizer, Parallel work of full-text retrieval; No3. Section describes full-text retrieval Spark and Spark SQL Implementation plan generation and optimizer related concepts and principles; No4. Section describes the overall design of the Distributed Retrieval framework. Summarize the functions and functions of each layer; No5. Section describes the framework of SQL Grammar, Design and Implementation of retrieval parallelization and retrieval Optimization; No6. Section describes the framework application Spark SQL Situation and Spark SQL Overview of kernel modifications. No7. The performance and scalability of the framework are; Final Summary full text, And point out the next research work.

2. Related work

Research work related to the framework can be divided into relational interfaces for Big Data Processing Systems, SQL Execution Plan build and optimizer, Parallel Full-text retrieval. 3. Aspects.

1) Big Data Query Language and framework

Based on MapReduce, the big data processing system can give users powerful but low-level, Procedural Programming Interface. Programming Based on such systems is tedious, For efficient performance, User needs self-tuning. Therefore, Numerous Systems (Not limited MapReduce Programming Mode) Such as Pig [12], Hive [13], IMPA[10], Dremel[11], HBase[12] and Spark [14]. Provides Query Language Interface and Automatic Optimization Technology, No need to write low-level code and focus on low-level execution engine details, Enhanced User Experience.

2) SQL Execution Plan build and optimizer

SQL Execution Plan build and optimizer will SQL: The underlying execution engine can recognize the physical execution plan. Hive, Dremel, and Spark SQL all have Optimizer. Hive Via Antlr[15] Tool Recognition SQL Statement, And convert it to the underlying Mapreduce Task. Spark is through Catalyst[16] Engine parse. This kind of optimizer is done SQL Parse idiom tree, Turn it into an underlying execution task based on a series of analysis and optimization policies. Process is usually divided: Abstract syntax tree based on lexical and Syntax Parsing; Specification check; Get plan tree for metadata binding with data dictionary; Plan tree Optimization; Optimal Plan Selection; Physical execution.

3) Parallel Full-text retrieval

Full-text retrieval involves the establishment of index and quick query with index. The parallelization of full-text retrieval is the parallelization of these two processes in distributed environment. Full-text search engine Solr cloud and Elasticsearch The index is distributed to different machines through the fragmentation mechanism. When a search request comes, it is distributed to the machine where the index slice is located and executed in parallel, And returns the query result. Among them, Index operations on each machine are handled Lucene Done. The parallel process of multi-machine retrieval is communicated and scheduled by the system.

Currently, For Spark, The full-text retrieval tool for Spark is Cenerd[16], But spark-luceneIndex created can only be used in one job. Failed to reuse history index in multiple jobs. And no storage optimization for the index, Not available SQL Full-text search, Its group-oriented Spark Familiar developer.

3. Related Concepts and Principles

3.1 Full-text retrieval

Full-text retrieval refers to the computer's Index program that scans documents, Index every word, Record the location and number of occurrences of the word, When a user queries, Search program based on the index built in advance to find, And feedback the query results to the user's retrieval method.

The full-text retrieval system is a software system that provides full-text retrieval services according to the full-text retrieval theory. The structure of the full-text Retrieval System1. Shown, Generally speaking, The full-text retrieval system needs the basic functions of indexing and querying, And Text Analysis, External interface module.

1) RDD The Concept

RDDs a fault tolerant, Parallel Data Structure, Allows users to explicitly store data to disk and memory, And can control the partition of the data; Simultaneous, RDD also provides a rich set of interfaces to manipulate this data. RDD Interdependence to form Directed Acyclic Graphs (DAG), Spark Through analysis Dag Divide Task
Scheduling and execution, and provide Cache Mechanism to support data sharing during multiple iterations, greatly reduces the overhead of repeatedly reading data between iterative calculations. This is of great help to the performance of data mining and analysis applications that require multiple iterations.

2) Data Dependency and Performance

RDDs as data structure, is essentially a read-only collection of partition records. One RDD can contain multiple partitions, each partition is part of the data set. RDDs can be interdependent to form Directed Acyclic Graphs.

If RDDs only be one child per partition, the use of a partition, it's called Narrow dependence; if more RDD partitions can be dependent on, it's called wide dependence. Different operations may generate different dependencies depending on their characteristics.

Distinguish wide and narrow dependence on spark Job Scheduling and performance analysis are important. Narrow dependency means it can be done on the same machine; Pipeline operation equals to superimposing the operator of the data operation, avoiding more multitasking; And wide dependence is usually accompanied by dataShuffle operation, prone to performance issues. Therefore, good algorithm and frame design should avoid wide dependence.

### 3.2 spark sql Translation Engine

The Translation Engine is SQL. An important step in converting tasks that the underlying distributed computing engine can recognize.

In sparkSql China, okay. SQL the parsing process is through Catalyst Conducted, Catalyst's universal SQL translation engine, responsible for planning and optimization. The parse process is:

1) Use Antlr Grammar Parsing.

2) Parser Via Visitor Pattern will Antlr The formed syntax tree is replaced in Catalyst a plan tree consisting of Tree nodes defined in.

3) Analyzer Associating a plan tree with metadata information Optimizer Optimizes the plan tree, such as constant folding, Predicate Pushdown.

4) Physical Plan Converter (Spark Planner) is to convert each node of the plan tree to the underlying Spark Perform engine-matched physical plan. Each physical plan contains a pair RDD or the operation of the Data Source. RDDs Spark last step before Task Assignment and Scheduling. It represents the underlying data and the encapsulation of the Data Operator.

### 4. Frame Design

This article framework contains

#### 4.1 Layer

Receiving user query statements SQL Client layer, SQL Translation Engine, responsible for analyzing and optimizing the execution plan, Parallel Computing Layer, to provide distributed full-text retrieval, Distributed index storage layer, Negative SQL. The client can receive user input SQL statement, the query submission module will SQL Submit to Translation Engine, Results resolution module parses the query results and returns them to the client. Adopted Spark SQL Of CLI Implementation.

#### 4.2 Translation Engine

SQL The translation engine module is SQL Module that performs the plan tree and optimizes. Parse process similar to Compiler Principle, First of all, according to lexical and grammatical rules SQL Statement Segmentation, To form a grammar tree. The syntax tree contains a series of semantic actions on the table from the bottom up. After, Based on a series of replacement and optimization rules, Change the syntax tree structure, Via Costmodel Select the optimal physical execution plan tree for the execution engine and hand it over to the underlying execution engine, Execute result return SQL Client.

In sparkSql China, Catalyst As the execution plan build and Optimization Framework will SQL Statement resolution for multiple RDD Operation, RDD Formed Dag To the parallel computing engine for job planning and execution. SQL Interpreter adopts Catalyst, Through modification, Catalyst Recognition full-text retrieval Grammar, And maps full-text retrieval operations to include full-text retrieval capabilities SearchRdd Operation, And hand it to the parallel computing Engine.

#### 4.3 Parallel Computing Layer
Lexical, Syntax analysis, Indexing and other operations; The data source docking module writes the index of each partition data in the table to the distributed storage layer in parallel.; The parallelization of the query is based on the index parallel query of each partition. Finally through the Global Reduce Operation returns highest score K Results.

4.4 Distributed index storage layer

Distributed index storage layer adopts HDFS as a storage file system, the storage of the index is fragmented and replica. Thus, the concurrency and efficiency of job execution in parallel computing layer are improved. In this paper, two retrieval optimization schemes are designed and implemented. That is to say, the index storage and the original table data reduction strategy focusing on performance optimization and storage optimization are two scenarios. Total storage and index specified column policies; Also for storage optimization scenarios, Proposed time complexity O (N) Connection Algorithm Between query results and original table data. Full storage for scenarios where query results are obtained in the shortest amount of time. But the index storage is larger; The index-specified column works for scenarios with limited storage space. Index storage in the table has hundreds, thousands of column when the advantage very obvious. High Performance of data connection algorithm can assurance in can accept of time in return query results.

5. Each layer design

In big data processing and analysis system in SQLs user the operation of direct interface SQL Get a recognition user submitted SQL Query statement By internal definition of conversion and optimization rules Physical Implementation Plan. Physical implementation plan contains the how to implementation planning underlying job of details? Including operation of definition and your files system of interaction.

The conversion rules Support will query operation. Pushed to Data Source; In physical plan implementation module Implementation. SearchrddClass Contains index established and index query function. Which Grammar design reference MySQL. In the full text retrieval Grammar MySQL MySQL Yes 3 Of Style established Index. Here only reference a kind. This a kind of style compared with other two more in line with the user used.

5.1 Solution

This paper in grammar analysis module on Increase the full text retrieval grammar and the full text retrieval grammar recognition rules; In physical plan module on Increase the full text retrieval Grammar

The full text retrieval grammar of translation process as shown in Figure 3 Shown in SQL Statement by conversion for grammar tree final transformation for physical implementation plan Physical implementation plan contains RDD of Operation.

Column Information But because no storage corresponding of bIn stand-alone data less of situation under Storage strategy only need additional storage user need of Domain (Column) Information And not in storage and performance bottleneck problem. With the data of rise Situation will produce very big of difference. In massive data SQL Retrieval. Table usually contains do line or thousands of column of data. This produce the huge of additional storage overhead (InSpark And SOL R Or EST) The combined with "with in Also there are the problem). This a kind of difficult to "with simple of storage strategy.

Lack of Domain (Column) Data can by get the original table corresponding location of data to fill The query results by and the original table data associated find the missing data. Because associated operation of there Will produce corresponding of performance cost. Don't storage any column of data And only Index SQL In specified the need to index of Column And by associated operation get the Miss of data. This a kind of methods can effective reduce additional data of Storage But associated operation will reduce query performance.

5.2 Solution

Above the massive data retrieval in met of two class problem The storage and performance of comprehensive consideration. Based on this two class problem This paper put forward the two kind of storage strategy: Full amount of the storage strategy and Index (Token And Index) Specified column strategy. For different of reality demand This paper summarized and total. Two of storage strategy of application scene, Storage and index rules and the second kind of
strategy in associated Algorithm.

1) Applicable scene

The full amount of the index of strategy for an arcane^WithLuceneSupport random readTherefore corresponding domain of data can inO(1)Time get.

In only Index(TokenAndIndex)Specified column of Data Strategy inReturn results only contains score and documentID.Even though the can by documentIDAnd the document in the original partition data in offset consistent of characteristics find original dataBut most massive storage system for data of access the iterative the ModeDon't support random readCan't inO(1)Time in complete.

SoFor performance requirements is high sceneHope system can quickly return original dataThe full amount of the Index,For performance requirements don't high but data abnormal Pang

Partition alignment connection algorithm describe the has more partition of query results and has more partition of the original table the connection of Process.

Because for every original table partition establish the IndexAnd query for an arcane base on each index generation a query results of PartitionThe query results of each a partition is for the original table every a partition of query resultsSo partition form one by one mapping.Query results and original table wereRDDSaid.

Algorithm put forward of objective is through PartitionIDAnd offset(DocumentID)Find this twoRDDIn every mapping partition in the same of corresponding pointMakes the table data and query results data phase panelReturn contains the score and the original table related column of complete results.Algorithm complexityO(N),NFor Table data of total number of rows.

The algorithm of steps are as follows:

Steps1 Will query results and original table data of partition alignment(Based onMap, Partitions, with, index.)

Steps2 In query results of partition in"With dictionary record query results need to get of the original data of all offset and offset corresponding of score.

Steps3 In the original table of partition in by iterative and record offset of style find in dictionary in the offset Finally with the original data and score panel.

Steps4 Traversal all mapping PartitionUntil connection operation all complete.

6. sparksqlArchitecture and modify Overview

Among them,The bold part isSparkSQLKernel modified or added.System architecture andsparkSQLConsistent Architecture, Sharing4.Layer, Upper LayerSQLClient accepts user's SQLQuery, Translate EngineSQLAfter syntax analysis, Metadata binding, Plan Optimization, Physical Plan Transformation, Final conversion to pairRDD 'Here isSearchrdd' OperationExecution engine executionSearchrddRead-write and query operations for the partitioned index in,Among them,Index read and write are based onHDFSIIn parallel,Each partition forms an index.

7. Performance Analysis and scalability Experiment

This experiment uses10Physical Machine(1)TaiwanMaster, 9TaiwanSlave),Memory per physical machine is16 GB,CPUForIntel|R|Core(TM)|I7-2600CPU@3.40GHz8Core,HadoopVersion is2.7.1 SparkLatest Version for community-basedMasterBranch and join the branch version of the full-text retrieval module,Run inStandaloneMode,Maximum valid ClusterExecutor Number is36.

Set,This test set contains32440001Article document. For the experimental environmentSelect the beforeM,Document as an experimental dataDue to operation ability limitedM,iThe maximum value3243904,The number of documents and text form of space usage such as table5Listed in.

Performance Analysis:Because index of read and write and query are based on original dataTableOf partition parallelThe narrow rely onImplementation TimeTIndex performace Get globalTopKOf results need to two steps:In each partition in get the original number accordingWill score number of according to and the original number according to the spell pick upOf can

This paper from3A aspects to evaluation different storage strategy,Piecewise,The number of documents for system of influenceAnd index establishment time,The full text retrieval time and index Storage.
7.1 Establish Index

Experimental conclusion are as follows:

Combined with figure8And figure9,Analysis different document number of implementation time can foundSparkSQL.In full amount of the storage and index specified column strategy under established index of average time respectivelyMySQL Implementation Time0.6%And0.5%SoMySQL.It is difficult to adapt to the huge amounts data of the full text retrieval.

Combined with figure10And figure11,The different document number of implementation time analysis foundWhen data fixed whenWith the piecewise Number of increaseDue to more a task to be performed in parallelAnd the distribution to the amount of data reduceEstablish index of performance get improve.When piecewise number of fixed whenData2The index times riseImplementation time and the number of documents of slope less1.,(Retrieval of parallel effective relieve the by data

Problem:In piecewise number64 An arcaneGreater than cluster in mostExecutorQuantity35,Parallelism of which lead to the job need two-round to implementation completeEven though the each job distribution to the amount of data lessBut total job processing time increased(Include job planning,Factors such as startup).

As the number of documents increases,Although the number of slices has increased,But total job time approaches,The reason is that the space footprint of the Shard is greater than or equalBlocksize,Cause extra job start.

Index specified column policy only stores part of data,Saving a lot of disks?IoOperation,Compared to total storage policies,Average execution time reduced17%,But there is still a problem with insufficient Parallelism.

7.2 Full-text retrieval

The experimental conclusion is as follows:

Union chart12Tutu13,Analysis of execution times for different document numbers can be foundSparkSQL.The average execution time for full-text retrieval under full-volume storage and index-specified column policies isMySQLOF1%And10%

Union chart14, Tutu15, Perform-time analysis discovery for different document numbers,When the amount of data is fixed, As the number of slices increases, Query time reduced; When the number of slices is fixed, As the amount of data increases, Query time grew very slowly, So the framework has good scalability.

Partition alignment connection algorithm is required to get the original table data, Therefore, the index specifies that the column policy takes longer than the full storage policy..

Index specified column Policy, The decrease of index storage makes the number of partitions have a good relationship with the job execution time., In the current amount of data, No significant performance decline..

7.3 Index Storage

Union chart16Tutu17, The index storage under different document number is analyzed and found., Index the index storage for the specified column policy, yesMySQL Of 55.0%, Is full storage policy36.7%. Because only the necessary segmentation and index information is stored, Do not store original document, So as the amount of data and the number of columns increases, The advantage of index-specified column policies will be more pronounced.

7.4 Experiment Summary

Experimental results show that: Under the current experimental conditions, ContrastMySQL In total storage and index-specified column policies, The frame index build time is reduced to the original0.6%And0.5%, Query time is reduced to the original1%And10%, Index storage is reduced to the original under the index specified column Policy55.0%, And as the amount of data and the number of columns increases, Index quantity andMySQL The difference between the increase.

In conclusion, the traditional relational database system is difficult to meet the needs of full-text retrieval under massive data., However, the current mainstream Big Data Processing SystemSparkSimple data query and Analysis, There is no complete framework design and implementation for full-text retrieval.

This systemSQL Grammar Design, Retrieval Parallelism, Retrieval Optimization3. IntroducedSparkSQL Design and Implementation of
distributed full-text retrieval framework. In grammar Design, Support for indexing on several columns, Also provides a wealth of retrieval functions. To meet different retrieval requirements. In terms of index Parallelism, will be based on a single process/Thread-based full-text retrieval. Multi-node multi-task parallel processing. Effective solve the massive data under traditional database full-text retrieval of bottleneck problem. In retrieval Optimization. Put forward the two kind of storage strategy to deal with different scene. For performance and storage of requirements. And put forward O (N) Time complexity of partition alignment connection Algorithm. Finally By contrast different piecewise (Parallelism) And storage strategy under the index established. The full text retrieval, Storage Prove that the distributed full-text retrieval framework in performance, Index storage is far more than traditional relational database.

Next work will enhance the full text retrieval of Function Make its support dimension search and space search Optimization. Index piecewise strategy And will the contribution Spark Community.

References

2. Sun da wei Divergent opinions about YanZheng wei min. Big Data Flow Cytometry Calculation: Key technology and system instance
4. Xu Yihua, Zhou shengkui, Zhu qiuming, Wait. UAV communication channel Simulation and Based on flight trajectory J. Telecommunications Technology 20135(5):656-660
7. MAZ Shiz Zhou Etal. Design of signal space diversity based on non-binary LDPC code C. Internet communications. Fujian, China 2008 31-34.
8. Rongb Jiangt Li x Etal. combined LDPC codes overg f t With ar modulations for bandwidth efficient transmission. J. IEEE transactions on broadcasting 200854(1):78-84.
14. Songh Crujzr. Reduced-complexity decoding of qu-ary LDPC codes for storage magneticing J.
15. Ieetransactions on magnetics 20033(9):1081-1087.